UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

MEJORAS EN UN SISTEMA DE RECONOCIMIENTO DEL LOCUTOR DEPENDIENTE DE TEXTO

Álvaro Palomo Sánchez
Tutor: Doroteo Torre Toledano

Junio 2017

MEJORAS EN UN SISTEMA DE RECONOCIMIENTO DEL LOCUTOR DEPENDIENTE DE TEXTO

AUTOR: Álvaro Palomo Sánchez
TUTOR: Doroteo Torre Toledano

AUDIAS – AUdio, Data Intelligence And Speech
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2017

Resumen (castellano)

Este Trabajo Fin de Grado tiene como objetivo el estudio y la implementación de un sistema de reconocimiento del locutor dependiente de texto, empleando para ello modelos ocultos de Markov (HMM) como base de nuestro modelo acústico-fonético. Esto permitirá que con un mismo reconocimiento se pueda verificar la identidad del hablante mediante el análisis de las características de la voz y que la frase de paso pronunciada sea la correcta.

Para el desarrollo y experimentación de este proyecto se parte de la base y resultados conseguidos por el grupo ATVS en el TFG *Reconocimiento del locutor dependiente de texto: Experimentos con la base de datos RSR2015* de Álvaro Mesa, defendido en Junio del 2016, donde se consiguieron muy buenos resultados en verificación de frase mientras que los resultados en verificación de locutor fueron bastante más limitados.

Trabajaremos sobre el escenario de un sistema de acceso donde el sistema generará una frase aleatoria que el locutor tendrá que pronunciar.

El trabajo se basa en la verificación de locutor donde tenemos que ver el desempeño de nuestro sistema cuando un impostor dice la frase correcta, para ello usaremos la herramienta de reconocimiento de voz Kaldi y la base de datos RSR2015 parte III donde experimentaremos con secuencias de 10 dígitos pronunciados en lengua Inglesa.

Replicaremos los experimentos previos del TFG de partida, y a partir de ese punto intentaremos mejorar los resultados previos en verificación de locutor, donde había margen de mejora, y estudiar más a fondo como se comporta el sistema, analizando los scores por locutor y los scores individuales de las palabras.

Abstract (English)

The aim of this Bachelor Thesis is study and implement a text-dependent speaker recognition system, using Hidden Markov Models (HMM) as the basis of our acoustic-phonetic model. This will allow that with the same recognition can verify the identity of the speaker by analyzing the characteristics of the voice and that the pass-phrase pronounced is correct

For the development and experiments of this project we will use the base and results achieved by the ATVS group in the TFG Recognition of the text-dependent speaker: Experiments with the database RSR2015 of Alvaro Mesa, defended in June 2016, where they were achieved a very good results in phrase verification but the results in speaker verification were more limited.

We will work on the scenario of an access system where the system will generate a random phrase that the speaker will have to pronounce.

The project is based on speaker verification where we have to see the performance of our system when an impostor records the correct phrase, for this we will use the Kaldi speech recognition tool and RSR2015 part III database where we will experiment with sequences of 10 digits pronounced in English language.

We will replicate the previous experiments of the TFG and from here we will try to improve the previous results in speaker verification where there was improvements to do and to study more thoroughly how the system behaves, analyzing the scores by speaker and the individual scores of the words.

Palabras clave (castellano)

Reconocimiento del locutor dependiente de texto, Reconocimiento de voz, Kaldi, RSR2015, HMM, Modelos ocultos de Markov, fMLLR, MLLR, Adaptación al locutor, Modelos de locutor, UBM, ASR, Verificación de locutor.

Keywords (inglés)

Text-dependent speaker recognition, speaker recognition, Kaldi, RSR2015, Hidden Markov Models, HMM, fMLLR, MLLR, Speaker adaptation, Speaker verification, UBM, ASR, Speaker models.

Agradecimientos

En primer lugar me gustaría dar las gracias a mi tutor Doroteo Torre por confiar en mí para este proyecto, por haberme guiado durante el mismo, por el tiempo invertido y por prestarse en todo momento cuando lo he necesitado. Quiero agradecer también a Álvaro Mesa por su ayuda para retomar su trabajo.

Me gustaría dedicar especialmente este trabajo a mi familia, pero sobre todo a mis padres y a mi hermano que tanto me han apoyado estos años y han confiado siempre en mí, por su esfuerzo y por estar siempre ahí, sin vosotros no hubiera sido lo mismo.

Y por último me gustaría dedicárselo a mis amigos de toda la vida y a las grandes amistades que me llevo de esta carrera, sin ellos estos años de carrera hubieran sido muy distintos.

INDICE DE CONTENIDOS

1 Introducción 1
1.1 Motivación 1
1.2 Objetivos 2
1.3 Organización de la memoria 2
2 Estado del arte 5
2.1 Reconocimiento del locutor 5
2.1.1 Dependiente de texto 5
2.1.2 Independiente de texto 5
2.2 Funcionamiento de un sistema de verificación de locutor 6
2.3 Reconocimiento de voz 7
2.3.1 Modelos de lenguaje, léxicos y acústico-fonéticos 8
2.3.2 Extracción de características 8
2.3.3 Modelos ocultos de Markov (HMM) 9
2.3.4 Adaptación del locutor 11
2.3.4.1 MLLR y fMLLR 11
2.3.4.2 Adaptación MAP 12
3 Diseño 13
3.1 Herramientas utilizadas: Kaldi 13
3.2 RSR 2015 13
3.2.1 Organización de la base de datos 14
3.3 Punto de partida 15
4 Desarrollo 17
4.1 Protocolo de pruebas 17
4.2 Ficheros de train 17
4.3 Ficheros de test 18
4.4 Creación de los modelos de locutor 19
4.5 Fase de test 20
4.6 Calculo de scores 20
5 Integración, pruebas y resultados 23
5.1 Resultados previos 23
5.2 Resultados RSR2015 23
5.2.1 Exclusión de silencios en el score 24
5.2.2 Z-Norm 24
5.2.3 Análisis sobre la puntuación de las palabras 27
6 Conclusiones y trabajo futuro 31
6.1 Conclusiones 31
6.2 Trabajo futuro 31
Referencias 33
Glosario 35
Anexos ..... - 1 -
AResultados de verificación por palabra ..... - 1 -

INDICE DE FIGURAS

FIGURA 2-1: ESQUEMA BÁSICO DE UN RECONOCEDOR DE LOCUTOR [2] 6

FIGURA 2-2: BANCO DE FILTROS MFCC[6] 9

FIGURA 2-3: MODELO DE BAKIS [8] 10

FIGURA 3-1: COMPONENTES DE KALDI [7] 13

FIGURA 4-1: ENTRENAMIENTO MODELO DE LOCUTOR [1] 19

FIGURA 4-2: VERIFICACIÓN DE LOCUTOR[1] 20

FIGURA 5-1: RESULTADOS PREVIOS RSR2015 Y KALDI[1] 23

FIGURA 5-2: RESULTADO SCORE TOTAL SIN SILENCIOS 24

FIGURA 5-3: RESULTADO SCORE Z-NORM "A POSTERIORI" 25

FIGURA 5-4: RESULTADO Z-NORM 26

FIGURA 5-5: RESULTADO Z-NORM CON LA OTRA MITAD COMO COHORTE DE IMPOSTORES 26

FIGURA 5-6: RESULTADO CURVA DET CON PALABRAS IGUALES 27

FIGURA 5-7: RESULTADO RECONOCIMIENTO CON PALABRA ZERO 28

FIGURA 5-8: RESULTADO RECONOCIMIENTO CON PALABRA TWO 28

FIGURA 0-1: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA ONE ..... - 1 -

FIGURA 0-2: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA THREE ..... - 2 -

FIGURA 0-3: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA FOUR ..... - 2 -

FIGURA 0-4: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA FIVE ..... - 3 -

FIGURA 0-5: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA SIX ..... - 3 -

FIGURA 0-6: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA SEVEN ..... - 4 -

FIGURA 0-7: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA EIGHT ..... - 4 -

FIGURA 0-8: CURVA DET PARA RECONOCIMIENTO CON LA PALABRA NINE ..... - 5 -

INDICE DE TABLAS

TABLA 3-1: DISTRIBUCIÓN DE LOCUTORES EN RSR2015 [1] 14

TABLA 4-1: FICHEROS KALDI 18

TABLA 4-2: ESTRUCTURA FICHERO LATTICE-1BEST 21

TABLA 4-3: ESTRUCTURA FICHERO LATTICE-ARC-POST 21

TABLA 5-1: COMPARACIÓN EER ENTRE PALABRAS 29

1 Introducción

1.1 Motivación

Este TFG trata sobre el reconocimiento del hablante a partir de la voz. La voz es un rasgo biométrico que aporta muchísima información, a partir de la voz podemos conocer la identidad del hablante, edad, sexo, idioma o emociones entre otros.

La forma de comunicación más usada entre personas es el habla, si además juntamos el auge de las nuevas tecnologías donde podemos comunicarnos a distancia a través del móvil, telefonía fija o incluso Internet hace que la voz sea vital en nuestro mundo.

Por otro lado la debilidad de los controles de acceso tradicionales ha sido evidenciado, controles de acceso como contraseñas escritas hace que cualquiera que disponga de esta contraseña pueda acceder al sistema mientras que un sistema de acceso biométrico, como la voz, algo característico de cada persona, hace que no sea tan sencillo burlar estos accesos.

Cada vez son más las aplicaciones que se lanzan al mercado y que funcionan con la voz, por ejemplo los asistentes personales por voz, hacen que situaciones donde es incómodo manipular el móvil o cualquier dispositivo con la mano sean más fáciles, ahora bien estos asistentes personales se basan en reconocimiento de voz, y para que funcionen correctamente hay mucho trabajo detrás.

Existen muchas aplicaciones donde el procesamiento de voz es clave :

- Autentificación por voz: Este campo consiste en analizar la voz del hablante de manera que consigamos determinar si el hablante es la persona que queremos verificar.
- Reconocedor de voz: Aplicaciones que se encargan de transformar un segmento de audio en texto, de manera que funciona como un traductor voz-texto.
- Diarización de locutores: Consiste en la identificación de locutores dentro de ficheros de audio con el objetivo de etiquetar por segmentos que locutor interviene en cada momento.
- Análisis forense de la voz: El objetivo es lograr identificar a los locutores dentro de audios con fines judiciales donde puede ser la prueba de algún caso.

Este trabajo en concreto va a trabajar en la autentificación o verificación de locutores, donde un posible escenario sería el de un usuario que intenta acceder a un sistema protegido por acceso biométrico mediante la voz, este sistema le va a pedir al usuario que diga en voz alta su contraseña que previamente el usuario habrá seleccionado y entrenado para su uso personal.

Con el fin de evitar posibles grabaciones del locutor, se requerirá además que pronuncie una frase generada aleatoriamente por el sistema [3].

La principal motivación de este trabajo se basa en tratar de mejorar los resultados obtenidos para la verificación de locutor en un trabajo previo del grupo ATVS realizado por Álvaro Mesa, donde se consiguieron muy buenos resultados en verificación de frase, pero aún quedaba margen de mejora en la verificación de locutor.

1.2 Objetivos

Este TFG parte de la base de un TFG anterior del mismo grupo de investigación, en concreto del titulado *Reconocimiento del locutor dependiente de texto: Experimentos con la base de datos RSR2015*, realizado por Álvaro Mesa y defendido en Junio de 2016. En este proyecto se consiguieron unos resultados de verificación de frase muy buenos pero no tanto en verificación de locutor, donde había aún margen de mejora. Por eso nos centraremos en este último caso, en particular centrándonos en la situación en que el impostor dice la frase correcta y tenemos que discernir entre locutor o impostor únicamente por las características de la voz, y no por el contenido léxico de la misma.

Nuestro principal objetivo será, por tanto, el de mejorar la verificación de locutor conseguida previamente y realizar nuevas pruebas para analizar el comportamiento de nuestro sistema.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1: Introducción**

  Este capítulo contiene la motivación y los objetivos que nos hemos marcado en el TFG.

- **Capítulo 2: Estado del arte**

  Aquí vamos a tratar el estado del arte, las técnicas que se usan para reconocimiento del locutor dependiente de texto, la adaptación al locutor y el reconocimiento de voz.

- **Capítulo 3: Diseño**

  En este capítulo hablaremos sobre las herramientas que hemos utilizado así como la base de datos utilizada y nuestro punto de partida para el proyecto.

- **Capítulo 4: Desarrollo**

  Explicaremos como hemos llevado a cabo nuestros experimentos, desde la fase de generación de ficheros, pasando por la fase de test y train, y nuestra generación de scores. Hablaremos sobre las pruebas que vamos a realizar.

- **Capítulo 5: Integración, pruebas y resultados**

  Aquí mostraremos los resultados de nuestras pruebas expuestas en el capítulo 4 y veremos el rendimiento de nuestro sistema.

- **Capítulo 6: Conclusiones y trabajo futuro**

  El último capítulo se centrará en las conclusiones sobre los resultados obtenidos, si hemos cumplido con nuestros objetivos y veremos las líneas sobre las que se puede trabajar a futuro.

2 Estado del arte

2.1 Reconocimiento del locutor

Un reconocedor de locutor consiste en un sistema que es capaz de detectar la identidad del hablante a partir de un audio. Este sistema a veces también se encarga de reconocer las palabras dichas en este audio.

Hay 3 tipos de reconocimiento de locutor:

- Verificación o Autentificación: Este sistema consiste en que el usuario a identificar introduce previamente su identidad (identidad solicitada) y es el sistema, haciendo una comparación 1:1 el que se encarga de decidir si acepta o no la identificación.
- Identificación: En este tipo de sistema no existe previamente información acerca de la identidad del locutor, es el sistema el que se encarga de identificar al hablante haciendo una comparación 1:N, donde N es la cantidad de usuarios conocidos por el sistema.
- Detección: En este caso el sistema se encarga de detectar en segmentos de audio locutores, por ejemplo para la diarización de locutores (segmentar un audio según los locutores que intervienen en cada segmento), o detectar si un locutor está hablando en un determinado momento.

En el reconocimiento de locutor hay 2 tipos de reconocimiento en función de su dependencia con el texto de la locución, el reconocimiento del locutor independiente de texto y el reconocimiento del locutor dependiente de texto.

2.1.1 Dependiente de texto

El reconocimiento del locutor dependiente de texto, es el caso en el que el locutor dice una frase que el sistema conoce previamente, como puede ser una contraseña o una frase generada aleatoriamente y mostrada por pantalla que el locutor tiene que decir y el sistema solo tiene que comparar la voz grabada con el modelo de locutor de la identidad reclamada[3].

Estos sistemas se utilizan principalmente para sistemas biométricos con acceso a través de la voz. Donde el usuario reclama una identidad, introduce su contraseña mediante la voz y adicionalmente se puede pedir una frase generada aleatoriamente para evitar grabaciones.[3]

Nuestro proyecto se va a centrar en las técnicas para este tipo de sistemas y experimentaremos sobre este escenario.

2.1.2 Independiente de texto

El reconocimiento del locutor independiente de texto, a diferencia del dependiente de texto, el sistema no conoce previamente el contenido del audio a reconocer..

2.2 Funcionamiento de un sistema de verificación de locutor

El esquema de la figura 2-1 muestra la estructura de un sistema genérico de reconocimiento de locutor.

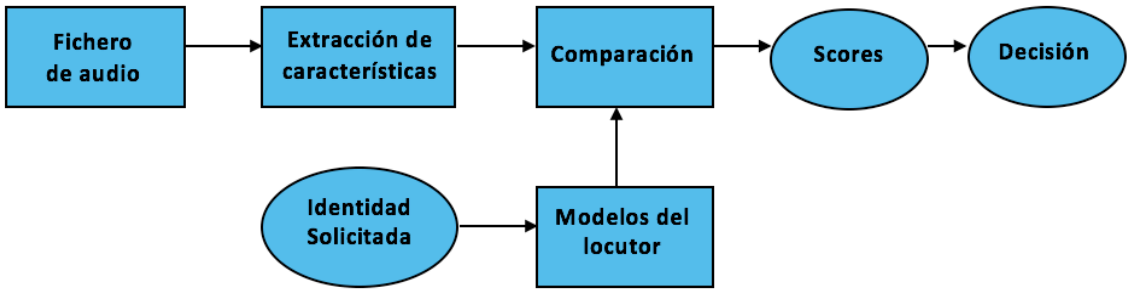

Figura 2-1: Esquema básico de un reconocedor de locutor [2]

En el esquema podemos ver que tiene 2 entradas, una entrada donde el locutor indica la identidad reclamada y otra entrada que es el audio de la locución. El objetivo final de este sistema será decidir si la identidad reclamada y la identidad del locutor es la misma, para ello pasará por una serie de bloques hasta llegar a la decisión, aceptar o rechazar.

El primer paso sería introducir la identidad reclamada, esto sirve para que el sistema identifique el modelo de locutor a utilizar y pueda hacer la comparación 1:1, ya que se trata de verificación.

El segundo paso consiste en grabar una locución y pasársela al sistema, la locución puede ser una contraseña o una frase generada aleatoriamente.

A esta locución le extraemos las características de la voz, usaremos MFCCs (Mel-Frequency Cepstrum Coefficients o Coeficientes Cepstrales en la Escala de Mel) que explicaremos en la sección 2.3.2, hay otros métodos para sacar las características, como los LPCC (Linear Prediction Cepstrum Coeficients o Coeficientes Cepstrales de Predicción Lineal), pero usaremos los primeros.

Estas características extraídas serán la entrada a nuestro bloque de comparación, la otra entrada será el modelo de locutor perteneciente a la identidad reclamada, previamente entrenado.

En el bloque de comparación, se hará una comparación de patrones (*pattern matching*), devolviendo un score o puntuación que usaremos para nuestra decisión.

En el bloque de comparación se hacen 2 tipos de comparación, usando 2 modelos distintos, el modelo universal de voz o UBM (Universal Background Model) y el modelo perteneciente a la identidad reclamada, generando 2 scores distintos.

El modelo UBM corresponde al modelo que llamaremos SI (Speaker Independent), que consiste en un modelo entrenado con una gran cantidad de locutores y locuciones, a partir de este modelo, que denominaremos a partir de ahora λ_{SI} , se generan unos scores, $Score_{SI.}$

El modelo de locutor es un modelo derivado del modelo UBM, pero que ha sido adaptado a las características particulares de la voz de un locutor con unas pocas locuciones de entrenamiento del locutor al que representa, mediante técnicas como fMLLR, explicadas más tarde. A este modelo, $\lambda_{SD,}$ le llamaremos SD (Speaker Dependent), y a partir de él se generaran unos scores, $Score_{SD.}$
Estos scores son probabilísticos e indican la probabilidad de que *X,* una locución*,* pertenezca al modelo con el cual se está comparando, se denota como :

$$Score_{SI} = P(X|\lambda_{SI})$$
$$Score_{SD} = P(X|\lambda_{SD})$$

Donde *X* sería nuestra locución de entrada.

A partir del cociente de estos 2 scores y un umbral Ω, podemos determinar si se acepta o se rechaza al locutor.

El score vendría representado por el siguiente cociente:

$$\frac{P(X|\lambda sd)}{P(X|\lambda si)} = \theta \begin{cases} \theta \geq \Omega \; Aceptamos\ al\ locutor \\ \theta < \Omega \; Rechazamos\ al\ locutor \end{cases}$$

Alternativamente se puede definir el score con la siguiente fórmula, aunque el umbral sería distinto[1].

$$\theta = Log(P(X|\lambda sd) \text{-} Log(X|\lambda si)$$

Con esta decisión podemos cometer 2 tipos de errores, llamados FA(Falsa Aceptación) y FR (Falso Rechazo)[5]:

- Falsa aceptación: Se considera falsa aceptación el caso en el que un impostor consigue acceder al sistema con una identidad que no es la suya.
- Falso rechazo: Se considera falso rechazo el caso en el que se rechaza a un usuario que reclama una identidad y esta identidad se corresponde con el locutor.

El umbral Ω, por tanto, lo controlaremos nosotros de manera que el error cometido entre falsas aceptaciones (FA) y falsos rechazos (FR) sea aceptable, a menudo se usa la Tasa de igual error, a esto se la llama EER (Equal Error Rate), donde el valor de FR es igual a FA.

El EER es una medida para evaluar el rendimiento de nuestro sistema de reconocimiento de locutor.

2.3 Reconocimiento de voz

El reconocimiento de voz se basa en reconocer a partir de un audio la transcripción de la locución. Para ello se utilizarán técnicas que veremos en los siguientes apartados.

Pero un reconocedor de voz no solo nos da una transcripción del audio, sino que también nos da información de las características del locutor, estas características las usaremos para crear un sistema de verificación de locutor.

2.3.1 Modelos de lenguaje, léxicos y acústico-fonéticos

Para crear un sistema de reconocimiento de voz hay que crear 3 modelos básicos sobre los que se asienta el reconocedor, el primero de ellos es el modelo de lenguaje.

El modelo de lenguaje nos indica cómo se forman las frases, es decir calcula las probabilidades que tiene una palabra ir seguida de otra palabra distinta o la probabilidad entre fonemas si bajamos un nivel. Estos modelos de lenguaje están basado en n-gramas y son esenciales ya que reducen la tasa WER (Word Error Rate) que indica el porcentaje de error de palabra, teniendo en cuenta borrados (B), sustituciones (S) e inserciones (I) de palabra frente al número de palabras reales (N) que aparecen.

$$WER = \frac{S+I+B}{N} \cdot 100$$

El segundo modelo básico es el léxico de nuestro reconocedor, está formado por todas las palabras que el ASR puede reconocer. Hay que tener un diccionario de base, y estas palabras estarán con sus transcripciones fonéticas, ya que es la unidad mínima de lenguaje y es sobre la que trabajaremos.

Por último el modelo acústico-fonético, este modelo es el más complejo, y se encarga de modelar como suenan los distintos fonemas dentro de una lengua. Para ello emplea modelos estadísticos creados a partir de una gran cantidad de locuciones. Se trabaja sobre los fonemas debido a que concatenando fonemas se pueden formar palabras y solo habría que buscar la secuencia más probable. Estos modelos acústico-fonéticos están formados por HMM que veremos en los siguientes apartados.[8]

Estos modelos acústico-fonéticos trabajan con unos vectores de características extraídos previamente en un proceso que explicaremos a continuación.

2.3.2 Extracción de características

Uno de los pasos esenciales en la creación de modelos fonéticos y en el reconocimiento de voz es la extracción de las características acústicas del fichero de audio. Para ello aplicaremos los **MFCCs**, es el método más utilizado para este propósito.

Los MFCCs (Mel Frequency Cepstral Coefficients) consiste en el uso de la escala de Mel como escala ajustada en frecuencia a las características de la percepción humana.

El proceso para la obtención de los MFCCs es el siguiente [6]:

1. **Pre-énfasis:** Consiste en aplicar una ganancia con el objetivo de compensar el efecto de la atenuación de altas frecuencias debido a la impedancia de radiación.
2. **Extracción de tramas**: Consiste dividir el audio total en segmentos de 25 ms, con un avance de 10 ms. Esto provoca un solapamiento del 40%.
3. **Enventanado**: A cada trama extraída se le aplica un enventanado que consiste en una multiplicación en el tiempo por una función ventana; la ventana de Hamming es la más utilizada.
4. **Calculo de los parámetros MFCC:**
    a. **Cálculo de la DFT:** Se calcula la DFT mediante la FFT para pasar la señal a frecuencia, quedándonos con el modulo del espectro de la señal.

b. **Cálculo de energía:** Calcularemos la energía aplicando un banco de filtros centrados en las bandas críticas en la escala de Mel. Este método simula el comportamiento del oído humano. Calcularemos la energía por banda.

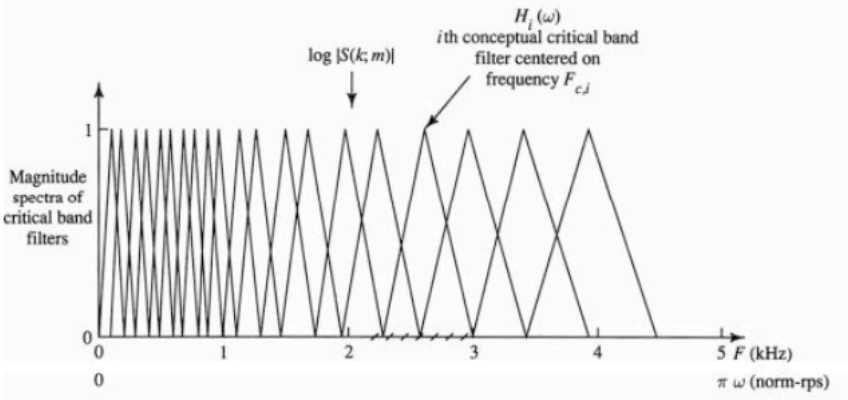

Figura 2-2: Banco de filtros MFCC[6]

c. **Transformación Cepstral:** Al resultado obtenido de la energía por banda se le aplica una DCT para obtener los coeficientes, de los que nos quedaremos con los primeros coeficientes, normalmente los 13 primeros coeficientes.

El resultado de estos coeficientes es lo que se llama como MFCCs, a estos coeficientes se les puede añadir parámetros como la velocidad o la aceleración, llamados delta-MFCC y delta-delta-MFCC. Son unos parámetros diferenciales que nos dan la variación respecto al tiempo.

Por otro lado, estos coeficientes MFCC pueden tener distorsión debido a los efectos de canal introducidos al grabar la voz del locutor, estos efectos pueden ser tanto lineales como no lineales. Por eso se aplican técnicas para compensar estos efectos, como pueden ser *CMS (Cepstral Mean Substraction)*, *Feature Warping o CMVN(Cepstral Mean and Variance Normalization)*.

La técnica de CMVN consiste en la normalización de los datos de entrada de manera que consigamos una distribución con media 0 y varianza 1.

2.3.3 Modelos ocultos de Markov (HMM)

Los modelos ocultos de Markov están basados en las cadenas de Markov. Un HMM es una cadena de Markov de primer orden y discreto, donde las salidas observables de los estados no se corresponden de forma determinista, si no probabilística. Son los modelos más utilizados en reconocimiento de voz, modelan estadísticamente la acústica de la voz. En la actualidad estos modelos están siendo parcialmente sustituidos por redes neuronales profundas, pero los modelos ocultos de Markov siguen presentes en la mayor parte de los reconocedores de voz.

En reconocimiento de voz, los estados se corresponden con los fonemas (parte inicial, media y final), y el objetivo es determinar la secuencia más probable.

Un HMM queda caracterizado por [8]:

- *El número de estados del HMM*, que denotaremos con la letra N. El número de estados es de nuestra elección.
- *El número de símbolos observables por estado, M*, usado para modelos discretos, en el caso de reconocimiento de voz la observación es un modelo continuo.
- *La distribución de probabilidades de transición, A:* Esta distribución nos indica la probabilidad de ir a un estado a otro. Es una matriz de probabilidades. Debido a que en reconocimiento de voz se usa un modelo de *Bakis* también llamado de izquierda a derecha, no podemos volver atrás en los estados, por lo que esta matriz tendrá probabilidades de transición nulas en algunos estados.

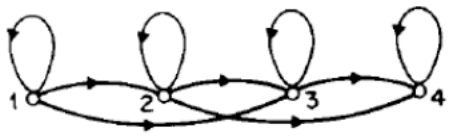

Figura 2-3: Modelo de Bakis [8]

- *La distribución de probabilidad de observación en cada estado, B:* Esto nos indica las probabilidades de las distintas observaciones en cada estado. Para el reconocimiento de voz se usa la función de densidad de probabilidad usando GMM. En muchos sistemas más modernos se emplean redes neuronales profundas para estimar estas probabilidades a partir de los datos.
- *La probabilidad inicial de ocupación de cada estado, Π.*

Para que los HMM fuesen útiles en aplicaciones prácticas se tuvo que dar solución a 3 problemas[8]:

1. **Dada una secuencia de observaciones y un modelo ¿Cómo calcular la probabilidad de observar la secuencia vista dada el modelo?**
   La solución al primer problema consiste en el algoritmo de *Forward / Backward,* este algoritmo se basa en definir una variable *Forward y Backward* e ir calculando las probabilidades de observar una secuencia parcial hasta el instante t y estar en el dado S_i en dicho instante. El punto clave es que se puede calcular la variable *Forward* del instante t+1 desde el instante t, gracias a la probabilidad de transición y de observación.
2. **Dada una secuencia de observaciones y un modelo ¿Cuál es la secuencia de estados que mejor "explica" las observaciones?**
   La solución a este problema se resuelve a través del algoritmo de Viterbi, con este algoritmo somos capaces de determinar la secuencia que mejor explica la observación. El algoritmo se encarga de recorrer todos los posibles caminos y ver en cada instante cual es el estado más probable.
3. **Dado un conjunto de observaciones de entrenamiento ¿Cómo ajustar los parámetros del modelo para maximizar la probabilidad de observar el conjunto de entrenamiento dado el modelo?**
   La solución del problema del entrenamiento consiste en aplicar el algoritmo de Baum-Welch, este algoritmo es un caso particular del algoritmo Expectation-Maximization o EM pero aplicado a los HMM. Consiste en maximizar una función que depende únicamente de los parámetros anteriores del modelo y de la nueva estimación del modelo, maximizar esta función nos lleva a una verosimilitud mayor en cada iteración.

2.3.4 Adaptación del locutor

La variabilidad fisiológica de las personas hace que sea una necesidad las técnicas de adaptar modelos específicos a cada persona. Pues cada persona tiene un timbre, forma del tracto vocal y órganos articularios diferentes.

En muchas de las aplicaciones de un ASR nos interesa discriminar locutores, y es por eso que necesitamos adaptar modelos de locutor.

En la mayoría de reconocedores de voz partimos de un modelo general llamado UBM que ha sido entrenado con una gran cantidad de locuciones de distintos locutores para conseguir un modelo lo más general posible.

Para nuestro caso particular tenemos que crear un modelo que se adapte a un locutor en concreto, se podría hacer de la manera que se crea un UBM pero necesitaríamos muchos datos del mismo locutor, algo que lo hace poco práctico. Es por ello que nacen técnicas para adaptar el modelo UBM a un locutor lo más fielmente posible. Las técnicas que vamos a ver son MLLR, fMLLR y MAP.

2.3.4.1 MLLR y fMLLR

La adaptación MLLR (Maximum Likelihood Linear Regresion) es una técnica que consigue reducir la cantidad de datos necesarios para adaptar el modelo. Se basa en entrenar una transformación lineal para transformar un conjunto muy grande de parámetros con una sola transformación.[8] Los parámetros que se ajustan de este modelo son las medias y las varianzas de las gaussianas (GMM).

Las medias de las gaussianas se ajustan mediante:

$$\mu = W\,\varepsilon$$

Donde W es una matriz W = [b A] , A es una matriz NxN que representa la transformación, b es un vector offset y ε es un vector de medias.

MLLR normalmente se aplica con un mayor número de clases de regresión después de aplicar un MLLR global e incluso también se puede aplicar una adaptación MAP que explicaremos en el siguiente apartado.

Las clases de regresión consiste en dividir en clases las distintas las GMM para hacer una transformación distinta en función de su clase. Esto aporta más flexibilidad ya que se pueden dividir las clases según fonema, por ejemplo.

Con este método se pueden conseguir buenos resultados con pocas frases cortas.

Una variante del método MLLR es el fMLLR, feature-space MLLR, es una transformación afín de la forma[12]:

$$x \rightarrow Ax + b \;\Leftrightarrow\; x \rightarrow Wx'$$

$$x' = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

En este caso la matriz W es la misma que para el caso de MLLR y cambia la matriz x'.

Ambas técnicas son ideales cuando tenemos poca cantidad de datos, además permite calcular la transformación solo con su matriz de transformación y es por eso que son las más usadas y son las que vamos a usar en nuestros experimentos.

2.3.4.2 Adaptación MAP

La adaptación MAP (Maximum a posteriori) o también llamada Adaptación Bayesiana, es una técnica que opera parámetro a parámetro sobre los HMM. Considera el modelo independiente de locutor como la información *a priori* y con la voz de adaptación estima dicho parámetro. También hay que fijar un parámetro de adaptación que nos indica un factor de adaptación a la nueva información[8].

El problema de esta técnica es que necesita mucha cantidad de datos para estimar correctamente la adaptación y requiere almacenar un nuevo conjunto de modelos completo.

3 Diseño

3.1 Herramientas utilizadas: Kaldi

La herramienta con la que vamos a realizar nuestros experimentos es Kaldi. Kaldi es una herramienta destinada para el reconocimiento de voz escrita en C++ y distribuida bajo licencia Apache 2.0, es una herramienta muy popular en ámbitos de investigación en reconocimiento de voz[7].

Kaldi es muy similar, aunque con diferencias, a otra herramienta que había sido ampliamente utilizada, llamada HTK, ya que esta nació con el propósito de hacer el código más moderno y flexible, aparte tiene un repositorio en GitHub donde se van implementando mejoras gracias a contribuciones de la comunidad.

Algunas de sus características son[7]:

- Integración a nivel de código de FSTs (Finite State Transducers).
- Soporte para operaciones Algebraicas basados en el estándar BLAS y rutinas LAPACK, que junto con los FSTs forman la base de Kaldi.
- Incorpora scripts genéricos de modo que puedan ser usados ampliamente en distintas bases de datos.

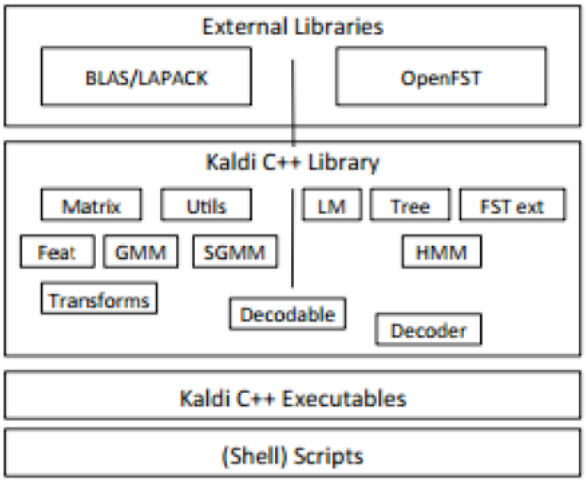

Figura 3-1: Componentes de Kaldi [7]

Los scripts implementados en la herramienta junto con sus funciones y modificaciones de las mismas será nuestra base para realizar los experimentos.

3.2 RSR 2015

La base de datos utilizada para la realización de los experimentos es la base de datos RSR2015.

RSR2015 está diseñada específicamente para evaluar sistemas de verificación de locutores dependientes de texto bajo distintos escenarios de pruebas. La base de datos se basa en hablantes ingleses con diferentes acentos que se pueden encontrar en Singapur.[4]

Algunas de las características de la base de datos RSR2015 son:

- Está formada por 300 locutores distintos, de los cuales 143 son femeninos y 157 masculinos, esto la convierte en una de las más grandes destinadas a la verificación de sistemas dependientes de texto.
- Un total de 151h de audio grabado en condiciones similares con 6 dispositivos móviles diferentes como tablets o Smart-phones.
- Varios grupos étnicos dentro de la base de datos.
- 3 partes diferenciadas con contenido léxico distinto y con el propósito de probar diferentes escenarios de reconocimiento dependiente de texto.
- Un protocolo de evaluación para cada una de las 3 partes de modo que haya una comparación justa entre sistemas.

3.2.1 Organización de la base de datos

La base de datos contiene 300 locutores diferentes y además está equilibrado en el número de locutores masculinos y femeninos, aparte contiene diferentes grupos étnicos, la siguiente tabla resumen lo refleja.

	Female	*Male*	*Total*
Chinese	118	119	79%
Malay	14	28	14%
Others	11	10	7%

Tabla 3-1: Distribución de locutores en RSR2015 [1]

Por otro lado, del total de las 151h de grabación se dividen en 9 sesiones, 3 de ellas se usan para la fase de train y las otras 6 para la fase de test.

La base de datos está dividida en 3 partes para recrear distintos escenarios de reconocimiento dependiente de texto, en cada escenario todos los locutores pronuncian la misma frase con el mismo contenido léxico para una sesión dada y solo la pronuncian una vez por sesión.

La parte I consiste en frases cortas para autenticarse, en cada sesión el locutor pronuncia 30 frases extraídas de la base de datos TIMIT. La duración media de cada grabación es de 3,20 segundos. Un ejemplo de frase es la siguiente:

Only lawyers love millionaires

La parte II consiste en comandos para un sistema de control doméstico. La duración media de cada comando es de 1,99 segundos. Un ejemplo de comando es el siguiente:

Watch movie

La parte III consiste en frases aleatorias de 10 y 5 dígitos, en cada sesión se generan 3 frases de 10 dígitos y 10 frases de 5 dígitos. Las secuencias de dígitos son distintas entre sesiones pero iguales para todos los locutores. La duración media por grabación es de 5,19 segundos y de 3,06 segundos respectivamente. Las frases son del tipo:

1 – 7 – 4 – 0 – 9 – 3 – 8 – 2 - 5 – 6
8 – 6 – 2 – 3 – 9

Para nuestros experimentos vamos a usar la parte III de la base de datos. Formado por frases de 10 dígitos que usaremos para la parte de train y de test.

3.3 Punto de partida

Como punto de partida en nuestro trabajo vamos usar el TFG *Reconocimiento del locutor dependiente del texto: Experimentos con la base de datos RSR2015*, realizado por Álvaro Mesa y defendido en Junio 2016, en este trabajo se realizaron experimentos con las herramientas de Kaldi y HTK y con las bases de datos de YOHO y RSR2015, pero únicamente nos vamos a quedar con la parte de Kaldi y RSR2015, ya que es lo más avanzado del TFG anterior y el punto de partida para este.

En los experimentos se siguió el siguiente protocolo de pruebas[1]:

- Se trabaja con la parte III de la base de datos RSR2015.
- De las 9 sesiones por locutor se usan 3 para train y las otras 6 para test.
- Con la fase de train se crea un modelo de locutor por sesión, con 3 frases de 10 dígitos por sesión.
- En la fase de test se sigue el protocolo proporcionado por la base de datos, se enfrenta cada frase de test a cada modelo de locutor creado.
- Se estudiaron 3 posibles escenarios distintos:

    - LOC-LOC: En este escenario se contempla el caso en el que el usuario target es verdaderamente el locutor que pronuncia la frase de test. Pero para esta situación hay 2 posibilidades, que la frase pronunciada sea la mostrada por pantalla en cuyo caso se corresponde con el caso target y la situación que la frase pronunciada sea errónea, que sería no target.

    - LOC-IMPOK: El segundo escenario corresponde al caso en el que sea cual sea el usuario la frase es correcta, pero existe la posibilidad de que quien ha dicho la frase sea un impostor, por lo que el usuario target corresponderá al locutor que dice bien la frase mostrada por pantalla y además es quien dice ser que es, mientras que el no target es el locutor que dice bien la frase pero no es el usuario que dice ser.

    - LOC-IMPNOOK: Para el último caso nos encontramos con la posibilidad de que el locutor que intenta acceder diga bien la frase y además sea quien dice ser que es, en cuyo caso es el target, y la posibilidad de que un impostor diga mal la frase mostrada por pantalla en cuyo caso es no target.

      En los 3 casos nuestro target va a cumplir las condiciones de que el locutor que pronuncia la frase es el verdaderamente el locutor en cuestión y además la frase es correcta.

En nuestros experimentos solo vamos a evaluar el caso LOC-IMPOK.

4 Desarrollo

En el primer paso en nuestro desarrollo vamos a preparar los archivos a partir de la base de datos, esta preparación de los archivos es fundamental ya que hay que pasárselos a las funciones implementadas en Kaldi. En segundo lugar tendremos que entrenar los modelos de locutor para crear un modelo de locutor por sesión. Usaremos estos modelos para enfrentarlos con los ficheros de test y con el resultado de la decodificación generar unos scores con los que podremos evaluar el rendimiento de nuestro sistema.

4.1 Protocolo de pruebas

El protocolo de pruebas a seguir es el mismo que en el TFG de punto de partida y el proporcionado por la base de datos RSR2015 que es el siguiente:

- Usaremos la parte III de la base de datos RSR2015, formada por frases de 10 dígitos, limitados en el rango de 0-10.
- Para la fase de train usaremos las locuciones de 3 sesiones {1,4,7} obteniendo un modelo de locutor por sesión mientras que para la fase de train usaremos las locuciones de 6 sesiones {2,3,5,6,8,9}.
- Cada modelo de locutor está formado 3 locuciones de 10 dígitos cada una.
- Para la fase de test enfrentaremos cada modelo de locutor con sus respectivos ficheros de test, la base de datos define estos emparejamientos en un fichero del que extraeremos las comparaciones para el test.
- De los 3 escenarios posibles LOC-LOC, LOC-IMPOK y LOC-IMPNOOK nos centraremos en el LOC-IMPOK, que consiste un enfrentamiento locutor/impostor en el que ambos dicen la frase correcta, nuestro target será aquel en el que el locutor es verdaderamente quien dice ser que es mientras que el no target es el locutor que no es quien dice ser que es.

4.2 Ficheros de train

El primer paso es crear los ficheros de test para adaptarlos a la entrada de las funciones de Kaldi.

La base de datos proporciona unos ficheros llamados *3seq10_dev_f.trn,* donde *dev* significa que pertenece al grupo de locutores de development, también hay un grupo de locutores de evaluation (*eval*) y la *f* que pertenece a locutores femeninos, aunque también hay masculinos, marcados con una letra *m*. Combinando ambas posibilidades tenemos 4 archivos.

Cada archivo referencia la sesión del locutor con sus 3 audios correspondientes a las 3 locuciones de 10 dígitos. El mapeo tiene la siguiente forma, (f/m)_locutor_sesión ruta_audio_1 ruta_audio_2 ruta_audio_3, un ejemplo es el siguiente:

f067_04 female/f067/f067_04_061.sph female/f067/f067_04_062.sph female/f067/f067_04_063.sph

Por otro lado tenemos el fichero *promptpart3.lst* donde tenemos un listado que asocia cada locución dicha en cada sesión con su transcripción.

Sabiendo esto generamos un script donde leemos estos 4 ficheros y creamos una carpeta por locutor y sesión, dentro de esta carpeta necesitamos 4 archivos esenciales que necesita la herramienta de Kaldi. Estos archivos son text, wav, utt2spk y spk2utt[7].

- El fichero de Text contiene la relación entre la locución o utterance y su transcripción.
- El fichero de wav contiene la ruta del fichero .wav de cada locución de esa sesión.
- El fichero utt2spk es un mapeo entre locución y locutor, este archivo es muy importante a la hora de tener una correspondencia entre locutores.
- Por último el fichero spk2utt es justo lo contrario de utt2spk, este fichero hace un mapeo entre locutores y sus correspondientes locuciones. Los 2 ficheros utt2spk y spk2utt son parecidos pero se usan en momentos diferentes.

Fichero	**Estructura**
Text	**<utteranceID> <text_transcription>**
Wav	<utteranceID> <full_path_to_audio_file>
Utt2spk	<utteranceID> <speakerID>
Spk2utt	<speakerID><utteranceID_1><utteranceID_2>…

Tabla 4-1: Ficheros Kaldi

Con estos ficheros por locutor y sesión tendríamos preparado todo antes de empezar la fase de train y la creación de los modelos de locutor.

4.3 Ficheros de test

El segundo paso es muy parecido a la preparación de los ficheros de train. Para la creación de los ficheros de test vamos a utilizar el fichero seq10_testdev_f, donde *dev* puede ser también *eval* y donde *f* también puede ser *m*, teniendo así un total de 4 archivos proporcionados por la base de datos, en estos ficheros se contemplan todas las comparaciones entre modelos de locutor y ficheros de test.

Cada fichero tiene una estructura en la que se indica el dígito indicado por pantalla, identidad a verificar y frase de test, tal y como se indica en el ejemplo.

1-7-4-0-9-3-8-2-5-6, f067_04, f067/f067_08_061.sph

En nuestro caso como nos vamos a centrar en el escenario LOC-IMPOK, que consiste en que la frase dicha por el locutor o por el impostor es correcta en ambos casos, nos fijaremos en que la frase mostrada por pantalla coincida con el audio, para esto tendremos que usar el fichero *promptpart3.lst* donde tenemos las transcripciones de cada fichero de audio.

A partir de estos 2 ficheros, con un script propio, se crearán los ficheros de text, wav, utt2spk y spk2utt, tal y como ya se hizo para la preparación de los ficheros de train.

Crearemos cada uno de estos 4 ficheros por cada modelo de locutor ya que habrá que enfrentar todas las frases de test con todos los modelos creados.

4.4 Creación de los modelos de locutor

Para la creación de los modelos vamos a usar las 3 frases por sesión extraídas en la fase previa de creación de ficheros de test. Tendremos un modelo por sesión de cada locutor, las sesiones que vamos a usar para la creación de los modelos son la sesión 1, 4 y 7.

Una vez que tenemos todos los ficheros de train generados estamos listos para empezar con la creación de los modelos de locutor.

El primer paso es extraer las características de cada audio, para ello vamos a emplear un script proporcionado por kaldi, *make_mfcc.sh*, este script nos calcula los vectores de características de los audios del locutor correspondiente.

Con los MFCCs calculados, aplicamos la normalización CMVN (Cepstral Mean Variance Normalization), esta técnica sirve para tener unas características del audio con media 0 y varianza 1, de esta manera conseguimos normalizar las características y conseguimos robustez frente a cambios en el canal y variabilidad en la grabación de los audios. Para ello aplicamos el script *compute-cmvn.sh,* también proporcionado por Kaldi.

Una vez que tenemos los MFCCs y hemos aplicado el CMVN, tenemos las características del locutor extraídas de 3 frases, procedemos a calcular las matrices de transformación por locutor.

Calcularemos una matriz de transformación por locutor y sesión empleando una adaptación fMMLR a partir de los datos calculados previamente, y calcularemos además una matriz de transformación masculino y femenino. Estas matrices de transformación las podemos obtener mediante el script de Kaldi *align_fmllr.sh,* obtendremos un archivo llamado *trans.1* que será nuestra matriz de transformación.

Con esto ya tenemos los modelos de locutor entrenados siguiendo el esquema descrito a continuación:

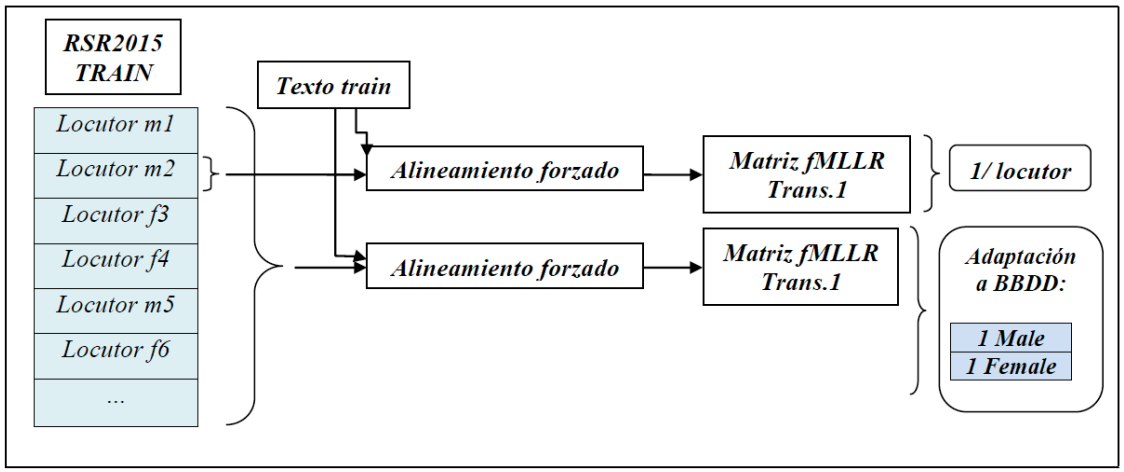

Figura 4-1: Entrenamiento modelo de locutor [1]

Como indica la figura 4-1, tendremos 2 matrices de transformación, una creada a partir de las locuciones de un solo locutor, que será el modelo de locutor y otra matriz de transformación generada a partir de todos los locutores de un mismo género. Estas matrices de transformación las usaremos después sobre el UBM.

4.5 Fase de test

Una vez que tenemos nuestras matrices de transformación a partir de la adaptación fMMLR iniciamos la fase de test. Para ello vamos a seguir el siguiente esquema:

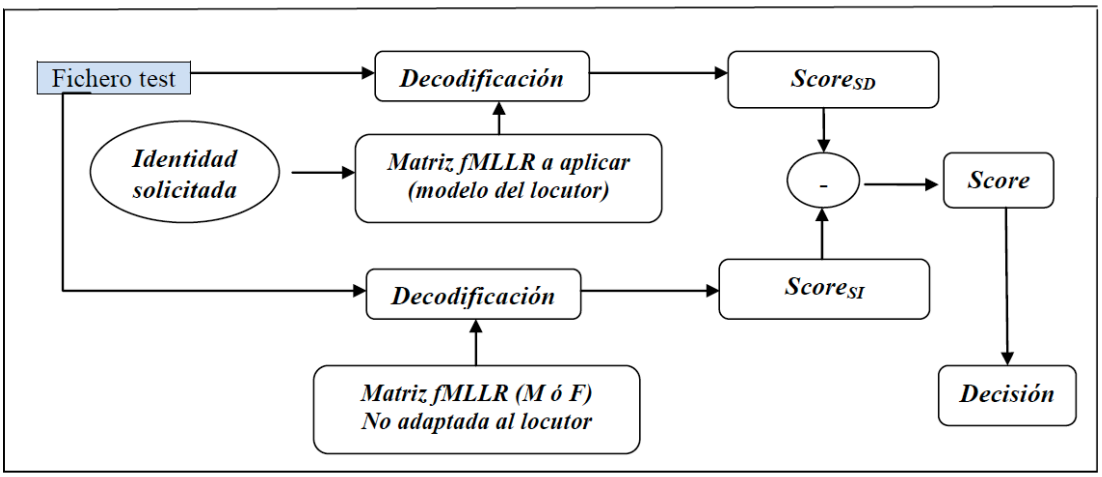

Figura 4-2: Verificación de locutor[1]

Haremos 2 decodificaciones no forzadas, una speaker-dependent a partir de las matrices calculadas por locutor y otra decodificación speaker-independent a partir de las matrices de transformación generales calculadas en función del género de la identidad solicitada.

Cada decodificación nos va a generar unos lattices, que es una representación en forma de grafo de las posibles secuencias de palabras que son suficientemente probables para una frase en concreto, en el siguiente capítulo se analizará su tratamiento para calcular nuestros scores.

4.6 Calculo de scores

Una vez que hemos hecho las 2 decodificaciones para todos los modelos de locutor y todos los ficheros de test tendremos un lattice por decodificación.

Nuestro objetivo es hacer un cálculo del score total por locución (log-like) basándonos en la puntuación acústica de cada palabra en la locución de test sin incluir la puntuación asociada a los silencios. En el trabajo que nos sirvió como punto de partida se hizo este mismo calculo pero sin excluir los silencios, y obviamente en los silencios no hay información acústica para discriminar entre locutores.

Para conseguir esto vamos a usar funciones implementadas en kaldi ya que habrá que tratar los lattices.

En primer lugar se hace un alineamiento del lattice, vamos a usar la función *lattice-align-words-lexicon*, esto nos sirve para alinear el lattice de tal manera que los arcos coincidan con las palabras.

En segundo lugar tenemos que calcular de nuevo los scores acústicos para colocarlos en sus lugares correctos debido al alineamiento previo, para hacer esto usaremos *lattice-scale –acoustic-scale=0.0* para poner las puntuaciones a 0 y después hacer un rescore acústico con la función *gmm-rescore-lattice*. Con este procedimiento tendremos las puntuaciones acústicas correctas y alineadas con las palabras.
Al lattice resultante vamos a aplicarle 2 funciones que nos generaran 2 archivos, uno con las puntuaciones acústicas por palabra y otro con el alineamiento de cada locución.

La primera función es *lattice-1best* que nos generará un archivo con las puntuaciones acústicas por palabra sobre el camino más probable de la decodificación.

f048_02_061	0	1	<eps>	7.99447	3617.52
f048_02_061	1	2	SIX	5.43243	2907.67
f048_02_061	2	3	ZERO	6.46429	2035.67
f048_02_061	3	4	<eps>	0	238.332
f048_02_061	4	5	EIGHT	2.78439	1540.38
f048_02_061	5	6	FOUR	7.98371	2078.71
f048_02_061	6	7	<eps>	0	1566.26

Tabla 4-2: Estructura fichero lattice-1best

Donde la puntuación del modelo de lenguaje es el primer valor y la puntuación acústica es el segundo valor.

La segunda función es *lattice-arc-post* donde calcula los posteriors y nos devuelve información relativa a cada arco. De esta manera vemos el número de frames por palabra.

f048_02_061	0	71	1	<eps>	SIL			
f048_02_061	71	44	1	SIX	S_B	IH_I	K_I	S_E
f048_02_061	115	36	1	ZERO	Z_B	IY_I	R_I	OW_E
f048_02_061	151	3	1	<eps>	SIL			
f048_02_061	154	25	1	EIGHT	EY_B	T_E		
f048_02_061	179	39	0.9999938	FOUR	F_B	OW_I	R_E	
f048_02_061	218	30	1	<eps>	SIL			

Tabla 4-3: Estructura fichero lattice-arc-post

La información por línea corresponde a *<locución><start_frame><num_frames><posterior><word><phones>*.

En nuestro caso como hemos generado esta información a partir del lattice generado por la función *lattice-1best* muchos de los posteriors son 1, esto es debido a que la función *lattice-1best* genera un lattice quedándose solo con el mejor camino de todos los posibles.

Ahora solo queda calcular el score acústico total excluyendo los silencios del cálculo. Este será nuestro score log-like. Tendremos 2 scores por locución, un score SD y un score SI. Para el cálculo final restamos el score SD al score SI, de manera que tendremos un score más negativo si el locutor "se parece menos" al locutor del modelo que al modelo universal y un score menos negativo si "se parece más" al locutor del modelo que al modelo universal.

5 Integración, pruebas y resultados

5.1 Resultados previos

Partimos de los resultados previos conseguidos en el trabajo *Reconocimiento del locutor dependiente de texto: Experimentos con la base de datos RSR2015,* TFG de Álvaro Mesa y defendido en Junio 2016.

Los resultados los mostraremos en una curva DET y nos fijaremos en el valor EER. La grafica siguiente muestra nuestro punto de partida para el caso LOC-IMPOK, teniendo en cuenta todos los locutores de la base de datos:

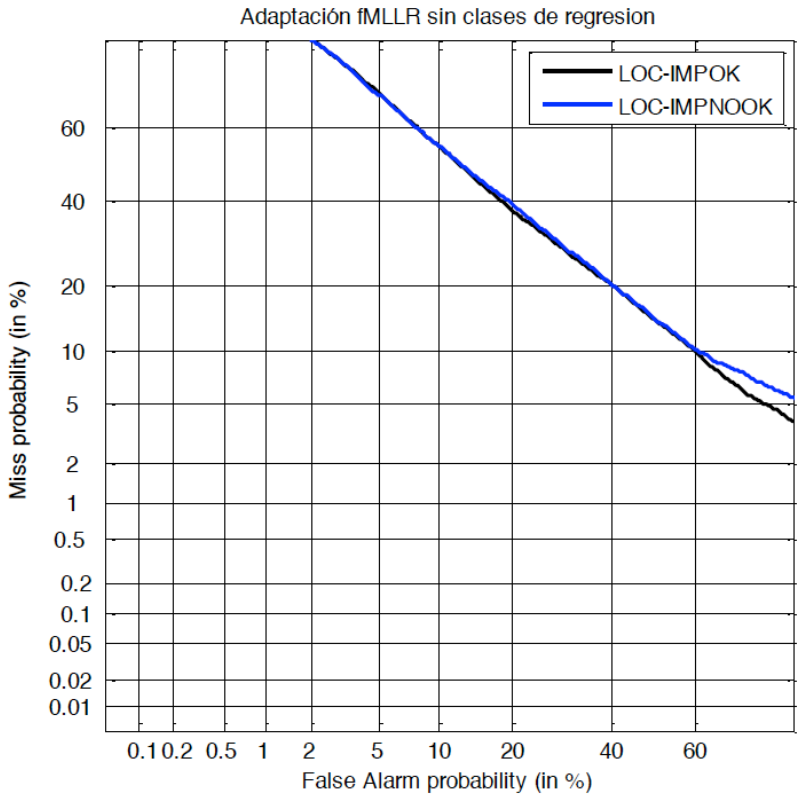

Figura 5-1: Resultados previos RSR2015 y Kaldi[1]

Como podemos ver el EER para el caso LOC-IMPOK está en torno a un 30%, en las siguientes secciones vamos a ir experimentando con diferentes pruebas con el objetivo de reducir la tasa de error.

5.2 Resultados RSR2015

Las siguientes secciones pertenecen a los resultados conseguidos en este trabajo.

5.2.1 Exclusión de silencios en el score

Nuestra primera prueba fue el cálculo de scores finales excluyendo los silencios, para el cálculo usamos todos los locutores de la base de datos, a continuación mostramos la curva DET correspondiente:

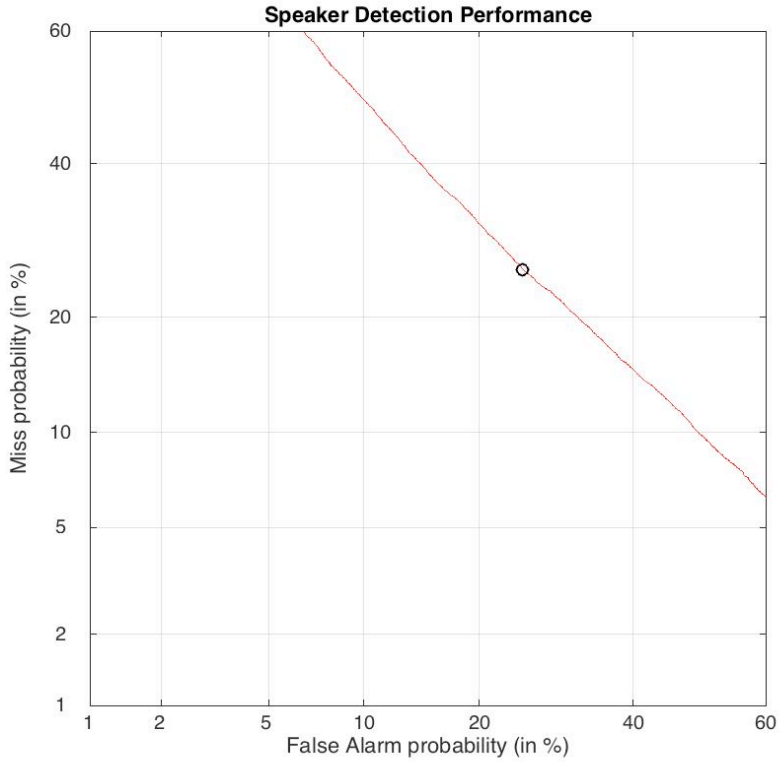

Figura 5-2: Resultado score total sin silencios

El EER está en torno al 25.8%, y vemos que se consigue una mejora importante respecto al resultado previo.

5.2.2 Z-Norm

Tenemos la hipótesis que no están alineados los scores obtenidos con un modelo de locutor respecto a los obtenidos con otros modelos del locutor, es por eso que vamos a aplicar una normalización de los scores llamada Z-Norm de manera que la distribución de scores de impostor tenga media 0 y varianza 1.

Debido a una posible desalineación de los scores por locutor vamos a aplicar una normalización a posteriori según:

$$Z = \frac{X - \mu}{\sigma}$$

Donde la X corresponde al score original, σ es la desviación estándar de los scores de impostor obtenidos comparando el modelo del locutor cuyos scores se quieren normalizar con una cohorte de normalización formada por impostores y μ es la media de dichos scores.

Tras aplicar esta normalización a partir de los scores de impostor y sobre todos los scores obtenemos la siguiente curva DET:

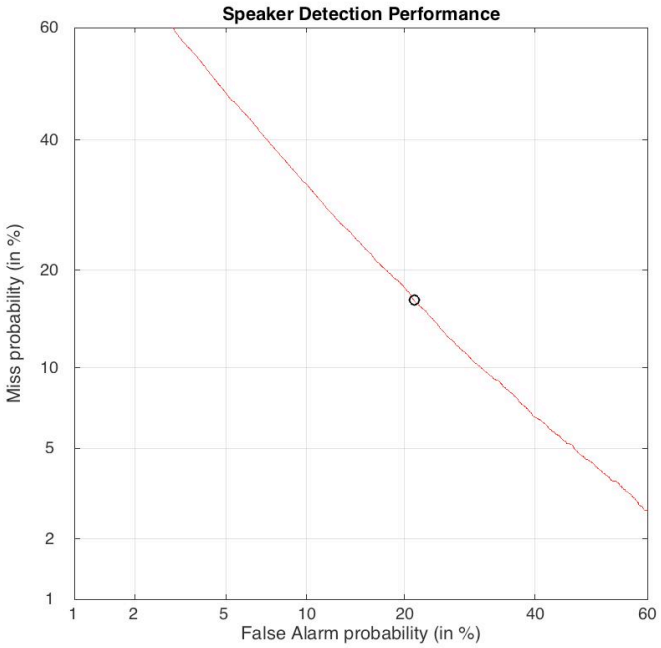

Figura 5-3: Resultado score Z-Norm "a posteriori"

El EER se sitúa en 18.9%, lo que significa que los scores sufrían un desalineamiento importante.

Esta primera prueba se realizó haciendo lo que denominamos una normalización "a posteriori", es decir empleando para estimar la normalización los scores de test de impostor. Como este método no se puede realizar en la práctica, una vez comprobado que hay desalineamiento, vamos a intentar aplicar un Z-Norm más realista.

Vamos a seleccionar dentro de los grupos de *dev* y *eval* que nosotros tratamos sin distinción, un subgrupo que será nuestra cohorte de impostores sobre la que calcularemos la media y la varianza para después aplicar la normalización sobre el resto de scores. De esta manera se puede aplicar la normalización siendo este caso más realista y práctico, pues un sistema real puede emplear una cohorte de normalización internamente para normalizar sus puntuaciones.

Seleccionaremos la mitad de locutores dentro del grupo de *dev* y *eval* como nuestra cohorte de locutores para "entrenar" la normalización, y aplicaremos esa normalización a la otra mitad.

Con este método obtenemos la siguiente curva DET:

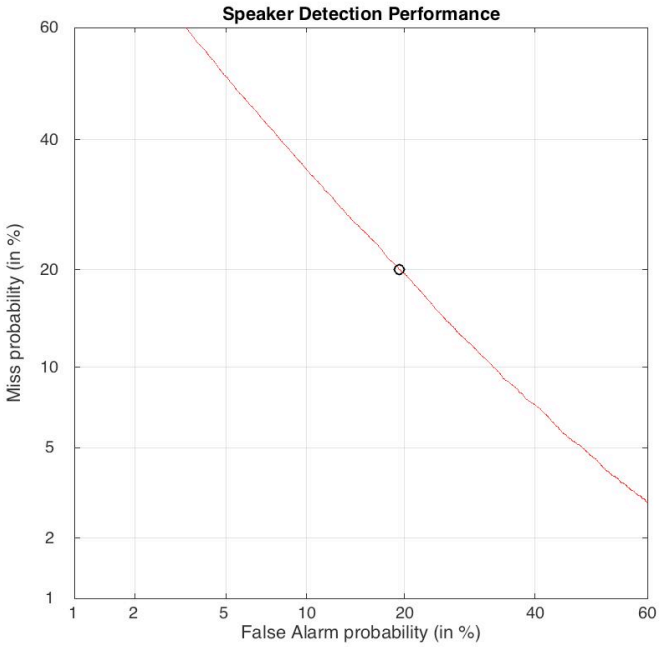

Figura 5-4: Resultado Z-Norm

Con este método el EER se sitúa en un 19.9%, la mejora global obtenida eliminando los silencios del cálculo y aplicando la normalización de scores con una cohorte de impostores sobre la que calcular media y varianza se sitúa en un 10% en términos absolutos y en un 50% en términos relativos. Es una mejora importante respecto al resultado previo que rondaba el 30% de EER. Cambiando la cohorte de impostores por la otra mitad conseguimos que el EER se sitúe en 19% igualando el score del Z-Norm "a posteriori".

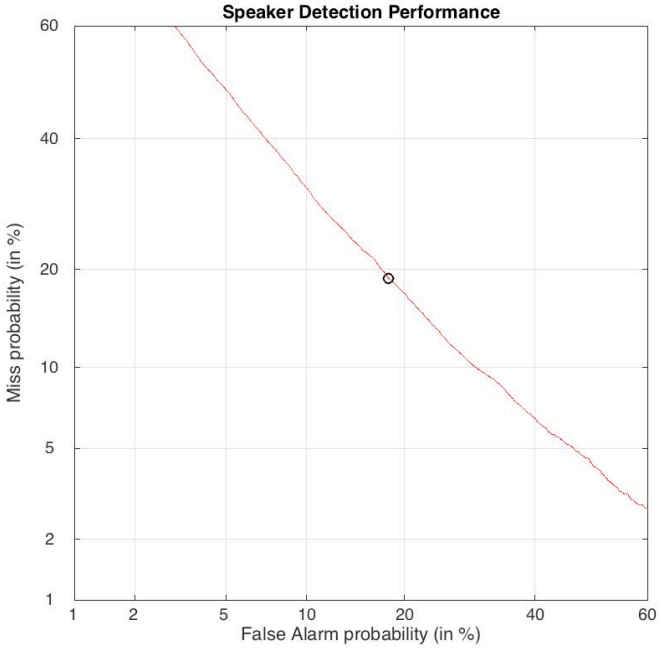

Figura 5-5: Resultado Z-Norm con la otra mitad como cohorte de impostores

5.2.3 Análisis sobre la puntuación de las palabras

Con los resultados anteriores hemos visto que hay una desalineación de los scores, y pensamos que puede ocurrir que el desalineamiento de scores se puede dar no sólo por locutor, sino también por cada una de las palabras que se pronuncian. Por ello vamos a bajar un nivel y en vez de actuar sobre la frase total vamos a actuar sobre las palabras individuales.

Nuestro primer paso será restringir las palabras sobre las que calculamos el score final. Debido a que es una decodificación no forzada en algunos casos se reconocen palabras distintas en la decodificación SI y en la decodificación SD.

Por ello vamos a calcular el score final teniendo en cuenta solo las palabras que han coincidido en ambas decodificaciones, tanto en palabra reconocida como en posición dentro de la locución.

Tras aplicar este procedimiento obtenemos la siguiente curva DET:

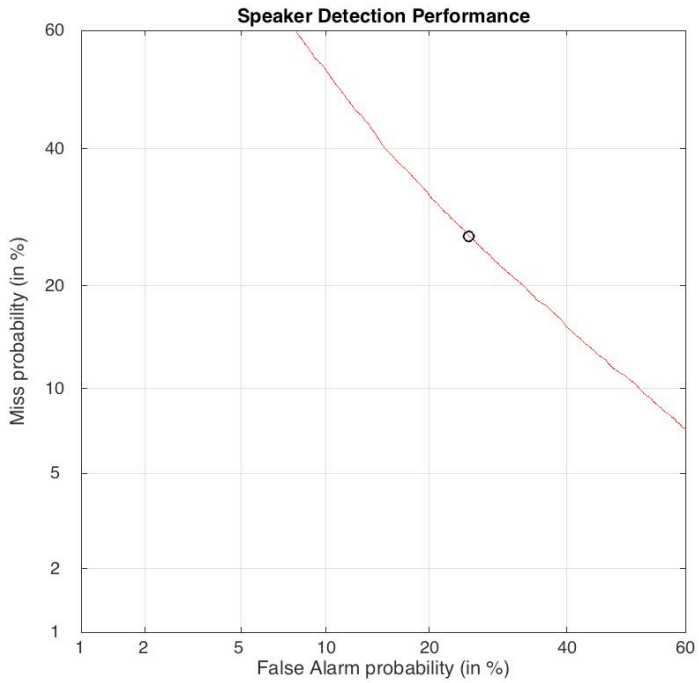

Figura 5-6: Resultado curva DET con palabras iguales

El EER se sitúa en el 26%, siendo muy parecido el EER obtenido solo con la exclusión de los silencios. Hay que destacar que en este punto todavía no hemos aplicado Z-norm.

Nuestros siguientes experimentos se van a basar en calcular los scores sobre una misma palabra (ZERO, ONE...) para todos los casos con el fin de determinar si hay palabras más discriminativas que otras y tratar de comprobar si hay palabras para las que el alineamiento de scores es peor que para otras, o si simplemente tienen distintos alineamientos de scores.

La siguiente gráfica representa la curva DET correspondiente al cálculo de scores utilizando únicamente la palabra *ZERO*.

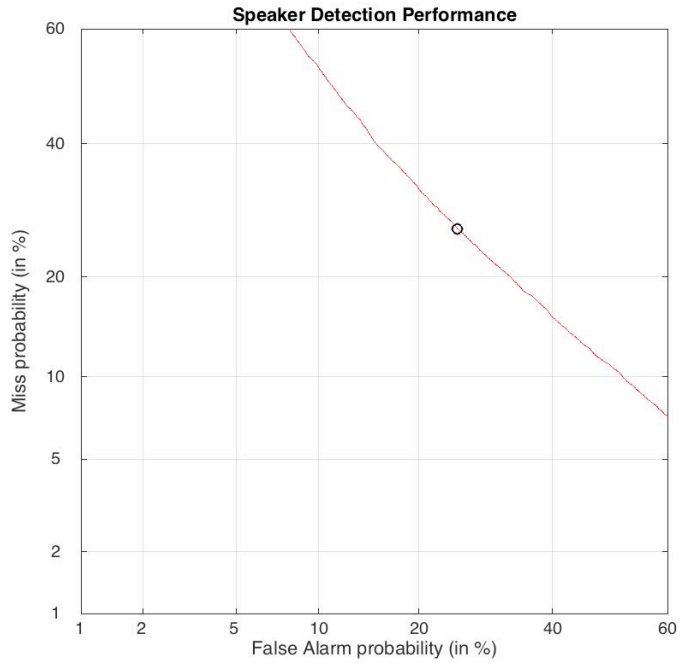

Figura 5-7: Resultado reconocimiento con palabra ZERO

Se obtiene un EER del 28%, muy parecido al que se obtiene con las palabras ONE, SEVEN, NINE.
Por otro lado la palabra TWO obtiene 35,5% de EER. La siguiente curva DET corresponde a la palabra TWO.

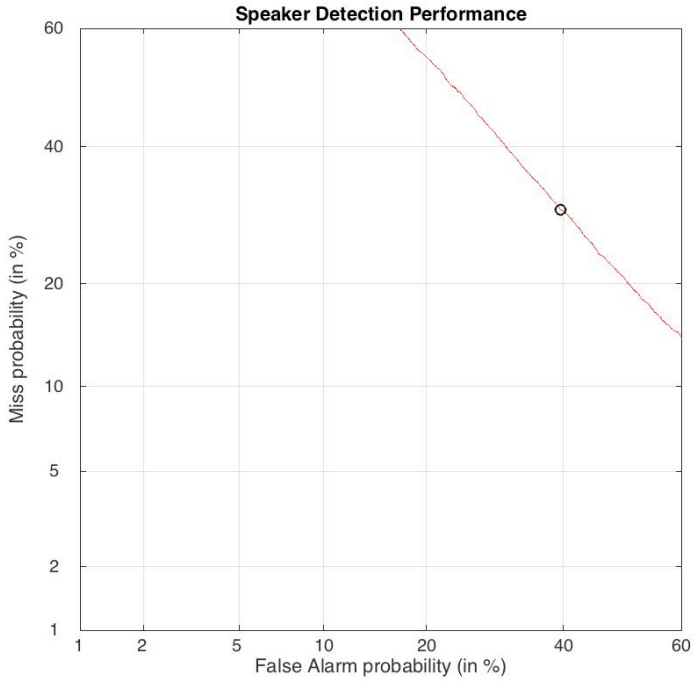

Figura 5-8: Resultado reconocimiento con palabra TWO

La siguiente tabla muestra una comparativa de EER entre palabras:

Palabra	EER
ZERO	27%
ONE	26,80%
TWO	35,50%
THREE	34,70%
FOUR	31%
FIVE	31%
SIX	36,70%
SEVEN	28%
EIGHT	33,90%
NINE	26,30%

Tabla 5-1: Comparación EER entre palabras

Por lo que se puede ver unas palabras el reconocedor funciona con un error mucho más bajo que otro.

Lo que resulta más llamativo es que con una única palabra ("ZERO", "ONE", "SEVEN" o "NINE") se consiguen resultados muy próximos a lo que se consigue con todas las palabras que coinciden. Esto lleva a pensar que si normalizamos adecuadamente los scores obtenidos por cada palabra y los fusionamos después podremos conseguir mejoras considerables [11].

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este trabajo se ha realizado un sistema de reconocimiento del locutor dependiente de texto usando como base el realizado en el trabajo *Reconocimiento del locutor dependiente de texto: Experimentos con la base de datos RSR2015,* realizado por Álvaro Mesa en Junio 2016.

Hemos usado la herramienta de reconocimiento de voz Kaldi y la base de datos RSR2015, orientada a experimentos de text-dependent.

Partíamos como resultados obtenidos previamente con un error del 30% , basándonos en EER, para la verificación de locutor. Como primera prueba decidimos ver el error cometido al calcular el score final excluyendo de este los silencios, pues los silencios no aportan valor a la decodificación, de esta manera conseguimos reducir el EER al 25,8%.

Como segunda prueba intentamos analizar que estaba ocurriendo con los scores, teníamos la hipótesis de que los scores al ser calculados por locutor no estaban alineados, es por eso que aplicamos una normalización a estos scores, llamada Z-Norm, para conseguir una distribución de scores con media 0 y varianza 1. Con esta prueba vimos que el EER bajaba hasta el 18,9%, por lo que nuestra hipótesis era correcta, pero al ser una prueba a posteriori no es útil en la práctica.

Como vimos que había una desalineación de los scores, intentamos aplicar una normalización de scores más real, para ello seleccionamos un grupo de locutores sobre los que calcularemos la media y la varianza, esto será nuestra cohorte de impostores. Una vez obtenidos una media y varianza por locutor, se normalizan los scores obtenidos para la prueba "real". De esta manera obtenemos un EER del 19,9% y 19% con lo que conseguimos bajar más aún el porcentaje de error para la verificación de locutor.

Debido a que se realiza una decodificación no forzada, en cada decodificación, una SI y otra SD, no siempre se reconocen las mismas palabras en ambos casos, es por eso que hicimos otra prueba que consistía en calcular el score final solo teniendo en cuenta palabras reconocidas que eran iguales en ambas decodificaciones, tanto en SD como en SI, y además coincidía en posición. Con esta prueba obtenemos un EER del 26%.

Decidimos ver la diferencia entre palabras individuales, con el objetivo de ver si algunas palabras eran más discriminatorias que otras, para calcular esto nos basamos en calcular el score final de cada palabra restringiendo la suma solo a una misma palabra común para todos los locutores y locuciones. De esta manera vimos que hay palabras que el reconocedor funciona mejor que para otras, palabras como ZERO, ONE, SEVEN o NINE funcionan mejor que para el resto. Y se puede ver que usando el reconocedor con una sola palabra se consiguen resultados parecidos a usar todas las palabras que coinciden.

6.2 Trabajo futuro

Como trabajo futuro nos faltaría analizar qué ocurre si normalizamos los scores por palabra previamente a sumarlos para componer un score final, teniendo en cuenta solo aquellas palabras que han sido reconocidas igual tanto en la decodificación SD como en la decodificación SI. Esta prueba a cierre de la memoria no se ha completado todavía.

También resultaría muy interesante fusionar los scores de palabra previamente normalizados, porque podría ocurrir que los resultados mejorasen respecto a los aquí presentados.

Por otro lado debido a que la base de datos RSR2015 es muy grande y proporciona varias partes distintas de pruebas como comandos, o reconocimiento para frases con 5 dígitos, aún quedan muchas otras pruebas que realizar sobre esta base de datos.

Como mejora para este sistema de reconocimiento del locutor dependiente de texto se podría estudiar la creación de un modelo acústico-fonético con el uso de DNN o redes neuronales profundas en lugar de los actuales HMM. El uso de DNN para los modelos acústico-fonéticos obtiene mejores resultados que los HMM en reconocimiento de voz, pero complica mucho la adaptación al locutor. En cualquier caso, hay esquemas en los que se usa como adaptación al locutor fMLLR anteriormente al uso de DNNs, por lo que el esquema en el que nos estamos basando seguiría siendo plenamente válido en estos casos.

Referencias

[1] Mesa Castellanos, Á. (2016). Reconocimiento del locutor dependiente de texto: Experimentos con la base de datos *RSR2015.* Madrid: UAM.

[2] Benesty, J., Sondhi, M. M., & Huang, Y. (Eds.) (2007). Springer handbook of speech processing, Chapter 37- Text-dependent speaker recognition. Springer Science & Business Media.

[3] González-Rodríguez, J., Toledano, D. T., & Ortega-García, J. (2008). Voice biometrics. In Handbook of biometrics (pp. 151-170). Springer US.

[4] Larcher, A., Lee, K. A., Ma, B., & Li, H. (2014). Text-dependent speaker verification: Classifiers, databases and RSR2015. Speech Communication, 56-77.

[5] D.Ramos Castro, Transparencias de la asignatura: Tratamiento de Señales de Voz y Audio: Introducción a la evaluación de sistemas de reconocimiento, pp.16-21, 2017.

[6] J.Ortega García, Transparencias de la asignatura: Tratamiento de Señales de Voz y Audio: Sistema Auditivo, Sensación Sonora y Parametrización Perceptual, pp.41-59,2017.

[7] Documentación online de Kaldi, http://kaldi-asr.org/doc/ (consultado el 21/05/2017).

[8] D. Torre Toledano, J. González Rodríguez, "Transparencias de la asignatura: Tratamiento de Señales de Voz y Audio: HMMs", 2017.

[9] Especificaciones de la base de datos RSR2015, https://www.etpl.sg/innovation-offerings/ready-to-sign-licenses/rsr2015-overview-n-specifications (consultado el 24/05/2016).

[10] C. Esteve Elizalde, Reconocimiento de locutor dependiente de texto mediante adaptación de modelos ocultos de Markov Fonéticos, Proyecto Fin de Carrera, Escuela Politécnica Superior, UAM, 2007.

[11] Hernández López, D., Torre Toledano, D., Esteve Elizalde, C., González Rodríguez, J., Fernández Pozo, R., & Hernández Gómez, L. (s.f.). T-norm y desajuste léxico y acústico en reconocimiento de locutor dependiente de texto.

[12] Povey, D., & Saon, G. Feature and model space speaker adaptation with full covariance Gaussians. Nueva York.

Glosario

GMM: Gaussian Mixture Model.

HMM: Hidden Markov Model

MLLR: Maximum Likelihood Linear Regresion

fMLLR: feature-space Maximum Likelihood Linear Regresion

MAP: Maximum A Posteriori

MFCC: Mel-Frecuency Cepstral Coefficients

LPCC: Linear Prediction Cepstrum Coefficients

CMVN: Cepstral Mean and Variance Normalization

CMS: Cepstral Mean Substraction

EM: Expectation-Maximization

EER: Equal Error Rate

FA: False Acceptance

FR: False Recognition

DNN: Deep Neural Network

DFT: Discrete Fourier Transform

FFT: Fast Fourier Transform

WER: Word Error Rate

UBM: Universal Background Model

TFG: Trabajo de Fin de Grado

ASR: Automatic Speech Recognition

Anexos

A Resultados de verificación por palabra

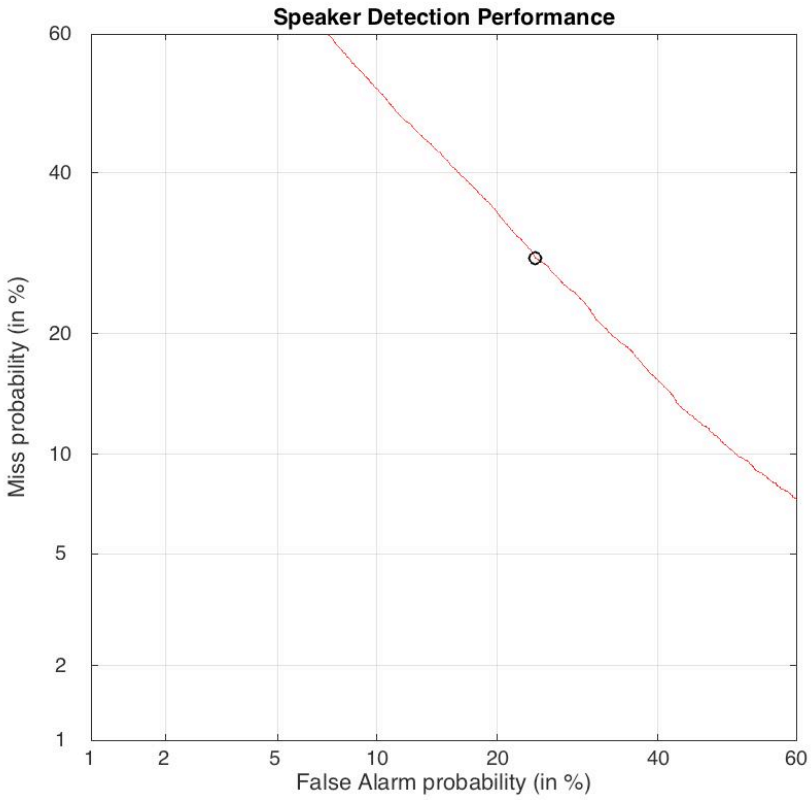

Figura 0-1: Curva DET para reconocimiento con la palabra ONE

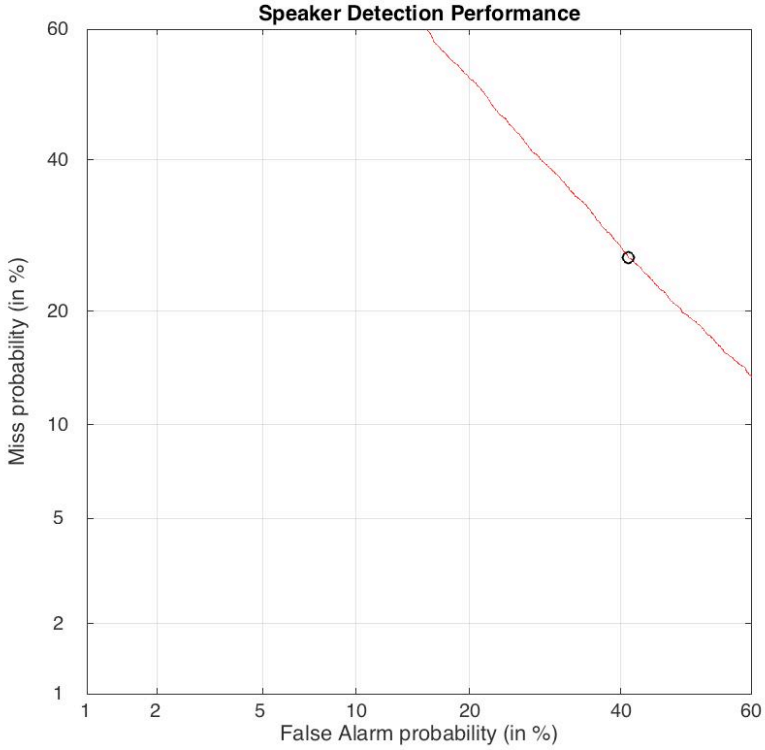

Figura 0-2: Curva DET para reconocimiento con la palabra THREE

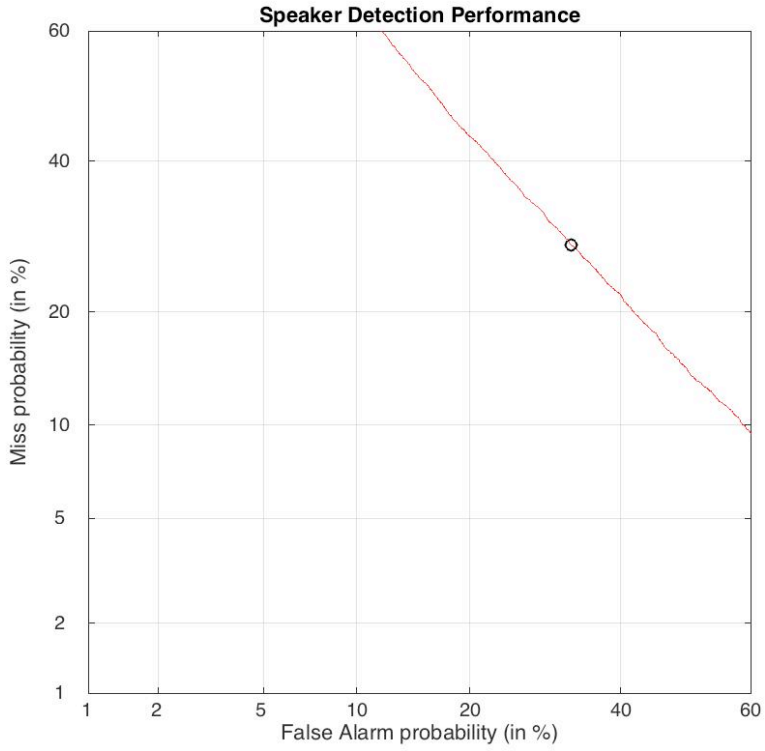

Figura 0-3: Curva DET para reconocimiento con la palabra FOUR

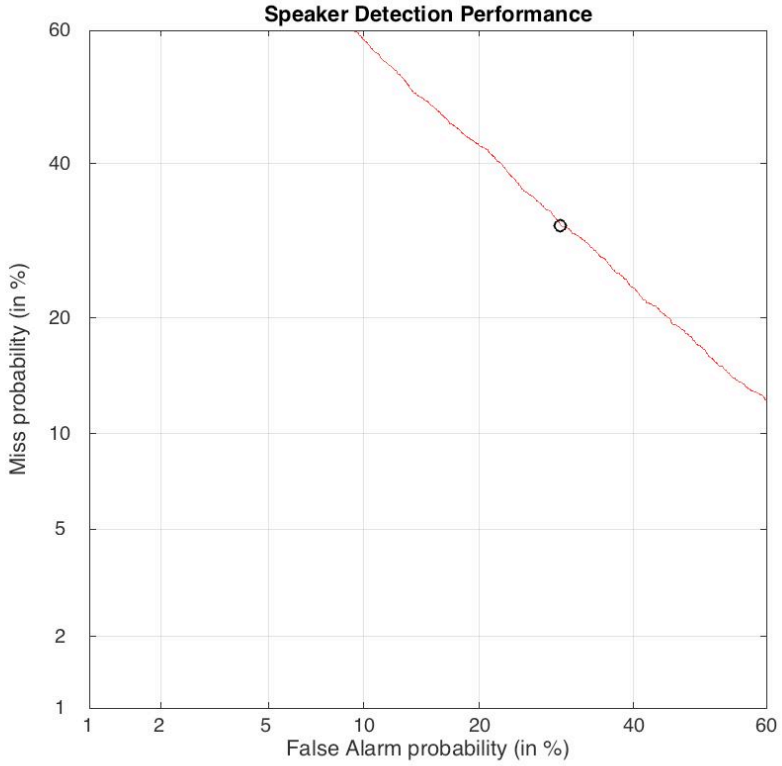

Figura 0-4: Curva DET para reconocimiento con la palabra FIVE

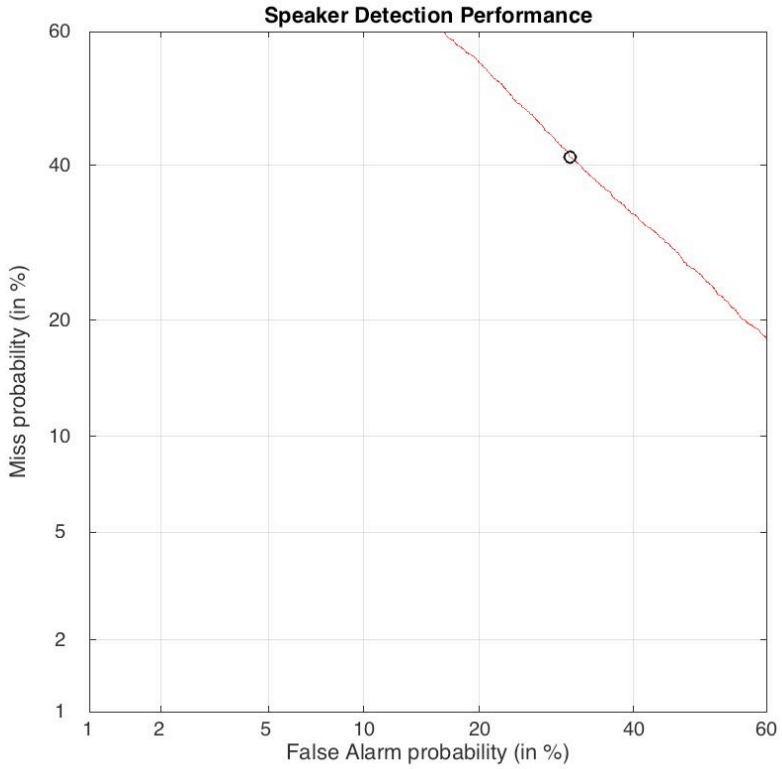

Figura 0-5: Curva DET para reconocimiento con la palabra SIX

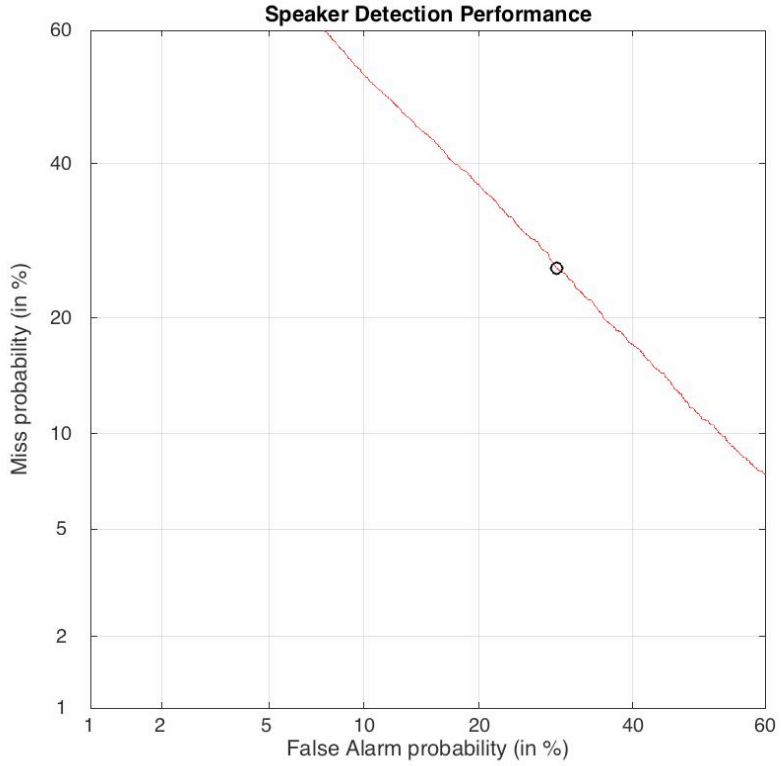

Figura 0-6: Curva DET para reconocimiento con la palabra SEVEN

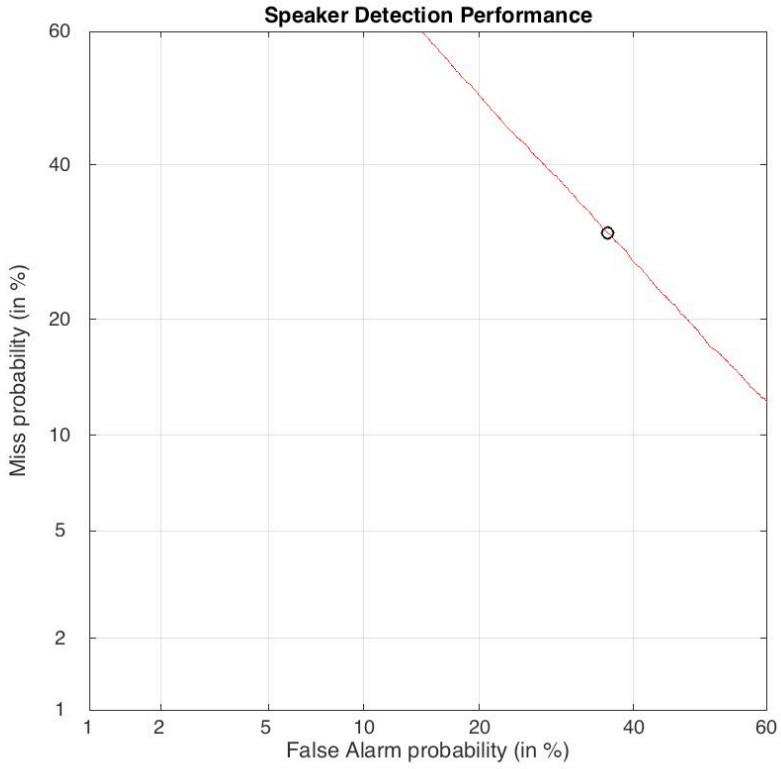

Figura 0-7: Curva DET para reconocimiento con la palabra EIGHT

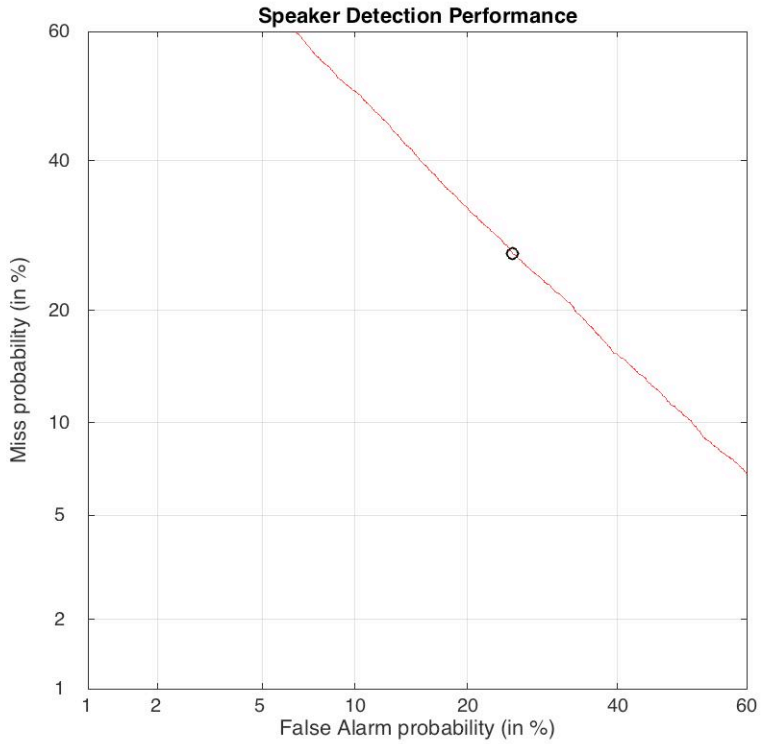

Figura 0-8: Curva DET para reconocimiento con la palabra NINE